International Journal of Computer Science and Application
IJCSA Volume 1, Issue 1, August 2012 PP. 19-25
ISSN(online)
ISSN(print)
www.ij-csa.org

Flexible Construction of High-Girth Qc-Ldpc Codes

Gabofetswe Malema
Department of Computer Science, University of Botswana, Gaborone, Botswana
Email: malemag@mopipi.ub.bw

(Abstract) This article presents a highly flexible method for constructing high-girth quasi-cyclic low-density parity-check (QC-LDPC) codes. The proposed algorithm constructs a Tanner graph formed by connecting groups of rows (check nodes) and columns (variable nodes) of the constructed code. To obtain a sub-matrix structure, rows and columns are divided into groups of equal sizes. Rows and columns in a group are connected in their numerical (positional) order to obtain a cyclic structure. Connected rows and columns must satisfy the desired minimum cycle length. We present conditions that guarantee desired girths. The proposed algorithm is by far more flexible in constructing a wide range (rates and lengths) of regular and irregular QC-LDPC codes compared to existing methods. The algorithm, which has linear complexity with respect to the number of rows or columns, provides an easy and fast way to construct QC-LDPC codes. Constructed codes show good bit error rate performances.

Keywords: QC-LDPC Codes; Tanner Graph; Girth; Code Rate and Length.

1. INTRODUCTION

Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) codes have been shown to be easily implementable in both encoder and decoder because of their block and cyclic properties [1]. They have also been shown to perform close to Shannon's capacity limit [2]. Performance of LDPC codes could be further improved by constructing codes with larger girths [3]. There are several methods for constructing QC-LDPC codes including algebraic, finite geometry, graphical and combinatorial techniques, examples of which are found in [4]--[8]. These construction methods avoid four-cycles by employing the row-column constraint. That is, no two rows share the same column more than once and vice versa. Although these methods can be used to construct a wide range of codes, they are generally limited in producing codes with arbitrary rates and lengths. They are also limited in their structure in that the number of sub-matrices and their configuration is fixed or limited in variability. The lack of flexibility constrains the variety of codes that could be constructed and may also constrain hardware implementation.

In [7] girth 12 QC-LPDC codes are obtain based on special graphs in which three connected vertices are used to form a column. The limitations of this method is that, it works with $(3,k)$ QC-LDPC codes, the sub-matrices sizes are not arbitrary and row and column weights are not always regular. QC-LPDC codes from finite geometries such as Euclidean (EG) and projective geometries (PG) are also limited in girth and code dimensions [6]. The same is true with QC-LDPC codes from Balanced Incomplete Block Designs (BIBD)[8]. The constraints for satisfying girths also involve use of prime numbers which restricts the size of sub-matrices. Other QC-LDPC codes construction methods such as one in [9] are also restrictive. The method guarantees girth 8 but restricts dimensions of obtained codes to multiples of $3k^2$xk^3, where k is code row-weight. One highly flexible construction method is found in [5] in which an arbitrary base or seed matrix could be used. A wide variety of rates and column and row weights can be used. The method sets algebraic conditions under which cycles of a particular girth are avoided for a given seed matrix. However, the conditions do not allow for all sizes of sub-matrices, p. A similar algorithm is found in [10] in which combinatorial designs are used to derive protographs that can result in girths higher than 12. This construction method also does not result in flexible codes and most obtained codes are also very large to be practically useful.

In this article we propose a method for constructing high-girth QC-LDPC codes. The method is highly flexible in that regular and irregular codes could easily be obtained over a wide range of rates and lengths. The number of sub-matrices could also be varied unlike in most previous methods. The method was initially proposed in [11][12] as a general method for constructing QC-LPDC codes by random searching. In [11][12] a distance graph is constructed that is then converted into a parity-check matrix. However, girths higher than eight could not be guaranteed for codes with column-weight of two and girths higher than six for codes with column-weights higher than two. We add conditions to the initial algorithm in [11][12] to guarantee higher girths. We also construct a Tanner graph instead of a distance graph. Although distance graphs are more compact than a Tanner graph, the algorithm complexity is higher than when a Tanner graph is constructed. Our algorithm works in a similar way to the methods in [4][13][14] in that we set the conditions to satisfy a given girth with given row and column weights then search for connections that satisfy these conditions. The advantage of our algorithm is that it works for any sub-matrix configuration or base matrix and could be used to obtain girths larger than 12.

This article is structured as follows. Section 2 presents the proposed algorithm for constructing flexible and high-girth QC-LDPC codes. Conditions for avoiding small cycles in a Tanner graph are introduced and illustrated. Bit-error rate performance analysis of obtained codes is presented in Section 3. Section 4 has concluding remarks.

2. PROPOSED ALGORITHM

Random or pseudo-random construction algorithms such as bit-filling (BF) [15] and progressive-edge growth (PEG) [16] have been developed to construct a wide range of codes. These algorithms construct a LDPC code by connecting rows and columns of a code one at a time provided a targeted girth and or rate is not violated. We take advantage of the flexibility found in random search methods such as BF and PEG to construct a wide range of QC-LDPC codes. To obtain QC-LDPC codes we add some modifications to these random search algorithms.

Our proposed algorithm has the following four main steps:

1) Divide rows (check nodes) and columns (variable nodes) of the constructed code into *j* (column weight) and *k* (row-weight) or more equal size groups respectively. The division of rows and columns into groups creates sub-matrices of the code.

2) The row-groups (RG) and column-groups (CG) are then paired (connections) such that each row-group appears *k* times and each column-group *j* times. The number of each row-group or column-group appearances (connections) determines the rate of the code. If the number of group appearances varies an irregular code is obtained. Row and column group connections form a base matrix or protograph.

3) For each row-column group (RCG) pair select a row, *i*, in the row-group, and search for a column, *x*, in the column-group that is at a desired distance (shortest path between nodes) from row *i*. Connect rows in the row-group to columns in the column-group according to the connection of row *i* and column *x*. That is, if row *i* is connected to column *x*, then row *i+a* is connected to column *x+a*. The connections are modulo of the size of row and column groups, p. These connections create a cyclic shift in the sub-matrices (shifted identity sub-matrices) of the constructed code.

4) Use the obtained Tanner graph to form a parity check matrix.

A (12,2,3) QC-LDPC code is constructed in **Figure 1** to illustrate the steps of the algorithm above. The number of row-groups is two (column-weight) and three (row-weight) for column-groups. Each group is of size four. Row-groups are denoted as RG and column-groups as CG. The groups are paired as[RG1,CG1],[RG1,CG2], [RG1,CG3],[RG2,CG1],[RG2,CG2] and [RG2,CG3] such that each row-group appears three (*k*) times and each column-group two (*j*) times. In each Row-Column group (RCG) connection a specified girth condition is observed. In

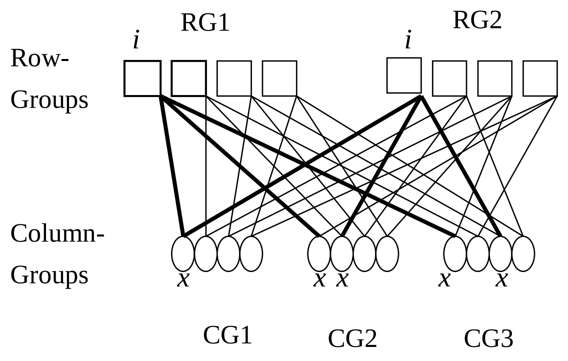

Figure 1. Construction of a QC-LDPC Code Tanner graph

	CG1				CG2				CG3			
	1				1				1			
RG1		1				1				1		
			1				1				1	
				1				1				1
	1					1					1	
RG2		1					1					1
			1					1	1			
				1	1					1		

Figure 2. A (12,2,3) QC-LDPC code in matrix form

I_s	I_s	I_s	I_s
I_s	I_s	I_s	I_s
I_s		I_s	I_s

I_s	0	I_s	0	I_s	0	I_s	0
0	I_s	0	I_s	0	I_s	0	I_s
I_s	I_s	0	0	0	I_s	I_s	0
I_s	0	I_s	0	I_s	0	0	I_s
0	0	I_s	I_s	0	I_s	I_s	0
0	I_s	0	I_s	I_s	0	0	I_s

Figure 3. Structure of obtained quasi-cyclic LDPC codes

this case four-cycles are avoided. Reference row *i* in RG1 is connected to the first columns labelled *x* in all the column groups and the connections are in bold. In each case the rest of the connections in RG1 follow that of row *i* and column *x*. In RG2 the reference row *i* is also connected to the same column *x* in CG1 and to the second and third columns labelled *x* in CG2 and CG3 respectively. The reference row *i* connects to columns *x* that do not form 4-cycles. The connections of RG2 are shifted by 1 and 2 in CG2 and CG3 respectively. The search for column *x* could be done sequentially or randomly. In sequential searches, a column-group is searched from beginning to end in sequential order. In random searches, a column is chosen arbitrary in a column group.

Figure 2 shows the matrix representation of the row-column connections of the code in **Figure 1**. The top row sub-matrices represent RG1 connections and the bottom sub-matrices RG2 connections. Column sub-matrices represent the three column-groups connections. When the number of row-groups and column-groups is equal to *j* and *k* respectively a code structure such as that of **Figure 3 (a)** is obtained where *Is* is a shifted *p x p* identity sub-matrix. When more than *j* row-groups or *k* column-groups are used, codes with zero sub-matrices are obtained as the example in **Figure 3 (b)** illustrates, where *O* is a *p x p* zero sub-matrix.

2.1 Avoiding Four-Cycles

In the four steps of the proposed algorithm outlined above, smaller cycles are avoided by connecting the reference row to a column that satisfies a desired distance. This condition guarantees girth of six if a column satisfying the distance of at least four is found. If the reference row *i* does not form a 4-cycle then all other rows in the same group do not form a 4-cycle. **Figure 4 (a)** shows how four-cycles are avoided to obtain girth of six. Column *x* must be at least a distance of four from row *i*. Since the connections of row *b* follow those of *i*, row *b* also avoids formation of 4-cycles. For row *i+a* (or row *b*) to form a 4-cycle, *x+a = z+a* in **Figure 4(a)**. That is, *x=z*. Since *i* is not connected to *z, (z!=x)*, then *i+a* also avoids 4-cycles. Also if *i* forms a six-cycle with *x, b* cannot not form a four-cycle with *c*. That would happen if *i+a = b* and *x+a=c*. That is, *x* is at a distance of 3 from *i* which should not be the case since we are searching for *x* with a distance of at least 4.

To obtain girth of eight, six-cycles are avoided. The girth-condition in the proposed algorithm is also sufficient for girth of eight. That is, if reference row *i* does not form less than eight-cycles then the rest of the rows in the group also do not form cycles less than eight. **Figure 4 (b)** shows how six-cycles are avoided by connecting rows and columns that are apart by at least six. Row *i* avoids connections to columns that are within a distance of six from itself. However, there is a possibility that smaller cycles could be formed when the rest of rows in the group are connected according to the reference row *i*. In **Figure 4(b)** column *z* is not connected to row *i* because it does not meet the distance condition of at least six. There may be a column *c* at a distance of three from row *i* and row *b* in the same group with *i* that is at a distance of three from column *x* (*i,b* and *c,x* in the same groups) as in **Figure 4(c).** If the difference (as indicated by arrows in the Figure) from row *i* to row *b* is the same as that from column *c* to *x* then row *b* will be connected to *c*. With *i* already connected to *x* an eight-cycle is formed between *i*, *x*, *b* and *c* as shown in **Figure 4(c).** These kinds of connections may result in formation of smaller than desired cycles. In this case, the smallest possible cycle formed is eight. Therefore, girth-eight is guaranteed if row *i* and column *x* do not form a six-cycle. Smaller cycles are formed by nodes connection of nodes on both sides. The smallest path length between these nodes is 3 and when connected they form a path of 8.

2.2 Avoiding Eight-Cycles

To obtain girth-ten, eight cycles are avoided. The girth condition in the proposed algorithm does not guarantee girth of ten. That is, smaller cycles could be obtained even if the cycle between row *i* and column *x* is at least ten. Smaller cycles could be formed as in **Figure 4 (c).** Although, row *i* and column *x* do not form an eight-cycle, if they are connected, they may map row *b* onto column *c* which leads to an eight-cycle between rows *i*, *b* and columns *x*, *c* as described earlier. These connections are made when the difference from row *i* to row *b* is the same as the difference from column *x* to column *c*. The differences are modulo of row and column groups' size, *p*. To avoid, formation of eight-cycles when targeting ten-cycles, the difference between row *i* and *b* must not be the same as that of between columns *x* and *c* given that the distance from *i* to *c* and *b* to *c* is 3. This is in addition to the condition that column *x* must be at the distance of at least eight from row *i* before they are connected.

2.3 Avoiding Ten-Cycles

To obtain girth-twelve in the Tanner graph ten-cycles are avoided. The reference row *i* must be connected to column *x* that is at a distance of at least ten. As in the case of girth-ten, smaller cycles could be formed when the rest of rows and columns are connected according to the connection of reference row *i* and column *x*. **Figure 5** shows the conditions that lead to formation of ten-cycles even when row *i* and column *x* do not form less than twelve cycles. Rows *i* and *b* are in the same group and so are columns *x* and *c*. If row *b* is at a distance of 4 from row *i* and column *c* is also at a distance of 4 from column *x* then the connection *i* to *x* and *b* to *c* forms a ten-cycle as in part (a) of **Figure 5**. In part (b) of **Figure 5** row *b* is at a distance of 3 from *x* and column *c* at a distance of 5 from *i*. In part (c) *b* is at a distance of 5 from *x* and column *c* is at a distance of 3 from *i*. If connecting *i* to *x* forces *b* to be connected to *c* then a ten-cycle is formed in each case. Therefore to avoid formation of ten-cycles, the conditions in **Figure 5** are avoided by ensuring that connecting *i* and *x* does not lead to a connection of *b* and *c* with the shown distances. The avoidance of ten-cycles is simply implemented by ensuring that the difference from row *i* to row *b* is not the same as that of from column *x* to column *c*. Eight cycles are avoided as described in subsection 2 above.

2.4 Avoiding Twelve-Cycles

To obtain girths higher than twelve, twelve-cycles are avoided. Twelve-cycles are avoided in a similar way to eight and ten-cycles. However, to obtain girths higher than twelve the base matrix must satisfy certain conditions as derived in [17]. Conditions for girths of 16, 18 and 20 are also derived for the base matrix.

Figure 6 shows conditions than lead to formation of twelve-cycles even when row *i* and column *x* do not form cycles less than fourteen. In part (a) of the Figure column *c* is at a distance of 5 from row *i* and *x* is also at a distance of 5 from *b*. If the difference from row *i* to *b* is the same of that from column *x* to *c* then *b* is connected to *c* which forms a twelve-cycle with *i* and *x*. A twelve-cycle is also formed when *b* is at a distance of 6 from *i* and *x* at a distance of 4 from *b* as in part (b) of the Figure. In part (c) row *b* is at a distance of 4 from *i* and column *c* at a distance of 6 from *x*. In part (d) row *c* is at a distance of 3 from row *i* and column *x* at a distance of 7 from *b*. Twelve-cycles are also formed when *c* is at a distance of 7 from *i* and *x* is at a distance of 3 from *b* as in part (e) of **Figure 6**. To obtain girth of fourteen these conditions are avoided.

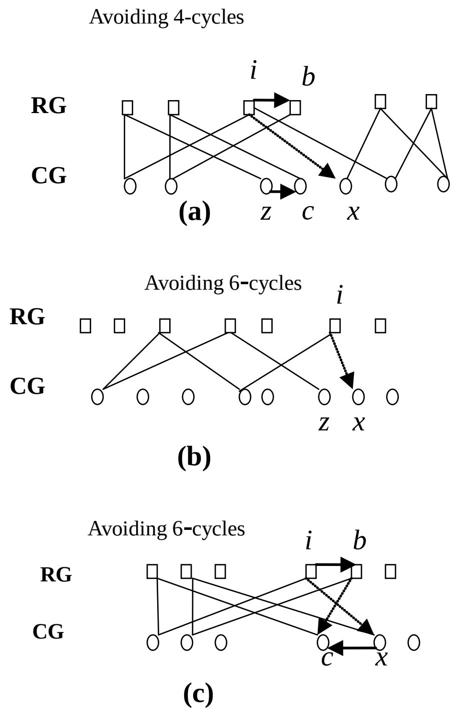

Figure 4. Avoiding 4- and 6-cycles in Tanner graphs

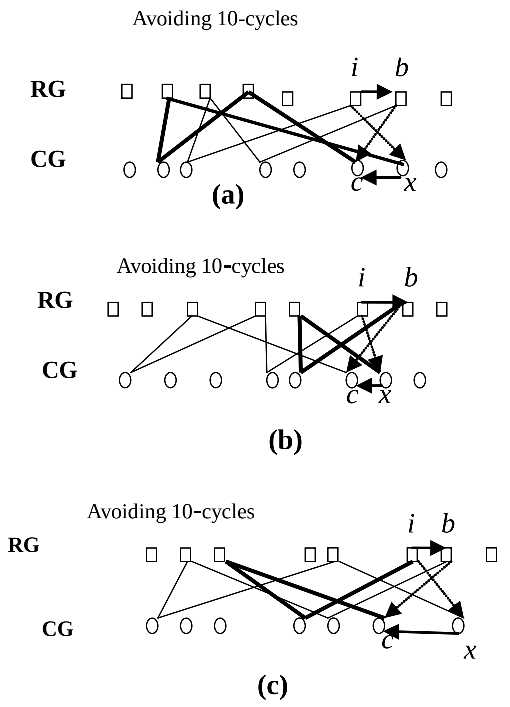

Figure 5. Avoiding ten-cycles in Tanner graphs

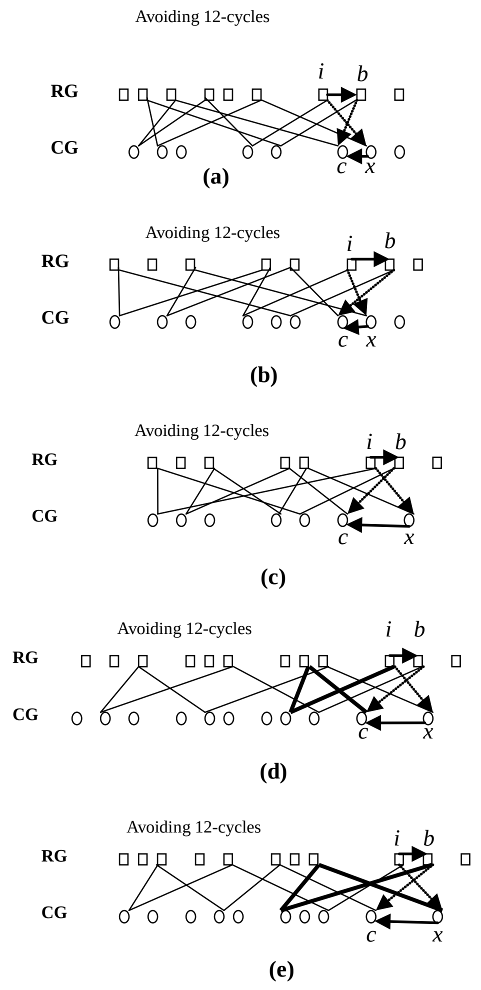

Figure 6. Avoiding twelve-cycles in a Tanner graph

2.5 Algorithm Analysis

The algorithm's conditions are that row i and x satisfy the desired girth and there are no connections of b and c that result in small cycles. The difference from i to b must not be equal to that of from x to c so that smaller cycles are not formed by b and c. Row-Column connections could be made either from the row or column side in the proposed algorithm. The algorithm is presented formally below with connections made from the row side.

QC-LDPC Code Search Algorithm

```
1) Divide code rows into $j'$ equal groups of size $p$, ($RG_1...RG_{j'}$) and columns into $k'$ groups of size $p$, ($CG_1...CG_{k'}$), where $j' \geq j$ and $k' \geq k$. k is code row-weight and $j$ is column-weight. $r_x$ is row $x$. $U_{rx}$ is a set of rows and columns within a desired distance from row $r_x$. $c_x$ is column $x$. $U_{cx}$ is a set of rows and columns within a desired distance from column $cx$. Distance is the shortest path between any two nodes (rows or columns).
2) Make row-column group (RCG) pairings according to your specifications such that each row-group appears $k$ times and each column-group $j$ times for regular codes. The number of such group pairings is $j'k$ or $k'j$ for regular codes. The row-column groups are ($RCG_1$ ...$RCG_{kj'}$).
3) For $t = 1$ to $kj'$ {
    Select $r_i$ from RG in $RCG_t$, where $1 \leq i \geq p$.
    Sequentially or randomly search for $c_x$ from CG
    in RCGt, where $c_x$ notin Uri and avoid smaller
    than desired cycles by using the difference
    condition. Else the algorithm
    fails.
    for $z = 0$ to $p-1$ {
        $r_{i+z}$ is connected to $c_{x+z}$. }}
4) Use the obtained Tanner graph to form a parity-check matrix.
```

One of the advantages of this algorithm compared to those in [11] [12][14] is that there is some pattern in the girth conditions. For example, avoiding eight-cycles the algorithm checks distances (path lengths) of 3 from row i and column x as in **Figure 4(c).** To avoid ten-cycles, rows and columns at distances of 3-5, 4-4, and 5-3 as in **Figure 5** are searched. For twelve-cycles the distances are 3-7, 4-6, 5-5, 6-4, and 7-3 as in **Figure 6**. The two distances add up to the avoided cycle length minus two and the minimum path length is three. Minus two because of the two connections from i to x and a to b are not yet counted. The algorithm works in a similar way regardless of the number and connections of row and column groups. The code rate is simply varied by changing the column and rows weights accordingly. The code length is varied by changing the size of sub-matrices, p.

The computational complexity of the algorithm is analysed and estimated in terms of the number of rows, M, and the number of row groups as follows.

- There are $j'k$ or $k'j$ row-column groups pairings for regular codes and each group is of size M/j'.
- Searching for a column satisfying the distance in each group takes M/j' operations. Testing for the difference conditions also takes about M/j' operations. Therefore each row-column group pairing takes about $2M/j'$ operations.
- Since there are kj' row-column group pairings for a regular code it takes $2j'kM/j'$ operations to complete all searches and connections. This is assuming the algorithm does not fail. The complexity of this algorithm is therefore O(M). However, this is a search algorithm and may fail many times even when it is possible to construct a code with some given parameters. Therefore, practically the algorithm may need many tries to obtain a code with given code parameters.

As shown by the complexity analysis above, the proposed algorithm computational complexity is linear with the number of rows, M. Although each execution of the algorithm does not guarantee that we will get a code (that is, algorithm may fail in step 3), our experiments show that the algorithm constructs a code most of the time even for small groups sizes with girths less than ten as shown by our MATLAB code in [18]. We easily obtained codes with minimum group sizes [4] for girth-six and eight codes. However, for minimum sizes of sub-matrices and higher girths (larger than eight) the algorithm may take a long time to find a code with sub-matrix sizes closer to minimum sizes.

2.6 Base Matrices with Zero Sub-Matrices

Codes with Zero Sub-Matrices: Matrices with zero sub-matrices are designed by using row and column groups larger than row and column weights. The number and interconnection of row-column groups may be dictated by decoding performance, hardware architecture or just be random. Codes with zero sub-matrices could be regular or irregular. In [19][20] protographs are used to design codes that match the structure of a semi-parallel architecture. A protograph is a small bipartite graph from which a larger graph can be obtained by a copy-and-permute procedure. The protograph is copied a number of times and then edges of individual replicas are permuted to obtain a single, large graph.

In [19] the protograph is designed to resemble decoder architecture. The decoder architecture may first be chosen based on a protograph decoding performance. Random or simulated annealing techniques were used to find the best performing protograph. Performance of the code also depends on the size of sub-matrices and their shift values. The matrix is expanded with p x p shifted identity sub-matrices. We could use the proposed algorithm to improve performance of constructed codes by improving their girths. The advantage of the proposed algorithm is that it can be used to construct codes with any sub-matrix or protograph configuration.

Irregular Codes: Carefully constructed irregular codes could have better performance compared to regular codes [21]. It was also shown in [22] that the degree of a variable node plays an important role in determining its error correcting performance. Variable nodes with a high degree tend to be decoded correctly compared to others. By targeting message bits to have higher degrees compared to parity bits we can improve the performance of a parity-check code. In [23] irregular quasi-cyclic codes from difference sets were found to have slightly better performance compared to regular ones. As stated in our algorithm, if the number of row or column groups is not uniform we obtain irregular codes. We can therefore construct irregular codes by having different numbers of appearances for row or column groups or both. A group weight is equal to the number of times it appears in connections. The proposed algorithm could be used to further improve performance of irregular codes by increasing their girths.

3. Performances Simulations

Bit error rate (BER) performances of constructed codes were simulated on an AWGN channel with BPSK modulation. Performance curves are shown in **Figures 7** and 8. **Figure 7** shows performance curves for (N,3,6) codes. Two new girth-ten (1998,3,6) QC-LDPC codes are compared against each other and a combinatorial QC-LDPC code [6]. One of the new QC-LDPC code is obtained with random selections while in the other code the searches for nodes satisfying the desired girth were sequential. Obtained QC-LDPC code using random selection performs better than the sequential QC-LPDC code. The results confirm what was found in [11][12] that sequential QC-LDPC codes perform worse that randomly constructed codes at girths of 6 and 8. However, in the case of girth-ten codes the performance gap is less than in girth 6 and 8 codes. This may be due to less regularity (pattern) in connections in higher girths compared to small-girths codes. The random QC-LDPC code also outperforms the QC-LDPC code from [6] and a girth-six random code of the same size at the shown simulation range. Also shown in the Figure is a (1008,3,6) code that slightly outperforms a code of the same size and girth obtained using the method in [9]. The new code performance better by about 0.05db at BER of 10-6.

Figure 8 shows performance of a new girth-14 (5776,3,4) code compared to a girth-12 code of the same size and rate from [7]. Our girth-14 code is outperformed by the girth-12 code. Also shown are high-rate QC-LDPC code performance curves. Our (3976,4,28) QC-LDPC code is slightly outperformed by a BIBD code[8] of the same rate and length in the shown range. At a rate 0.75 and code length of 4008, our code performs as well as a QC-LDPC code [4] of the same rate and length.

Obtained codes using our proposed algorithm perform as well as codes from other construction methods in shown simulation ranges. They do not show any performance superiority over other codes of the same size, length and girth. In some cases they are outperformed by codes with lower girths. It has to be noted that the structure of a LDPC code also plays an important role in its performance. In [4] the author proposed one way of searching for codes with better minimum distance. Other techniques such as avoidance of stopping sets [24] may be used to further improve performance of obtained codes. These techniques could be used in our proposed algorithm in future to search for better performing codes.

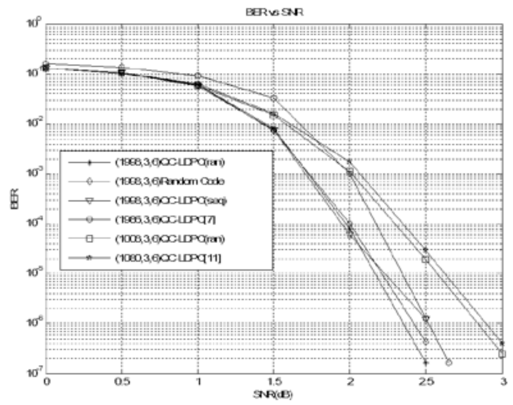

Figure 7. BER performance of regular (N,3,6) QC-LDPC Codes

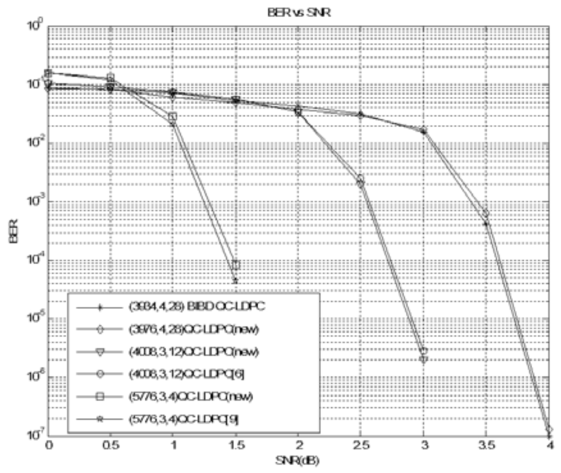

Figure 8. BER performance of high rate and girth QC-LDPC codes

4. Conclusions

A non-algebraic search algorithm for constructing high-girth quasi-cyclic LDPC codes has been presented. The algorithm is flexible in that, a wide range of rates and lengths can easily be obtained for both regular and irregular codes. The number and configuration of sub-matrices can also be changed to include zero sub-matrices. The algorithm is also simple to modify from one code design to another. For a given girth one needs to only change the group connections and size to obtain different code parameters. The algorithm offers more flexibility compared to previous developed QC-LDPC construction algorithms. Obtained codes show good BER performances comparable to random codes. Although obtained codes do not show BER performance superiority over other codes, they offer better flexibility in design.

REFERENCES

[1] S. Olcer, “Decoding Architecture for Array-code-Based LDPC Codes,” Proc. IEEE GLOBECOM, pp. 2046- 2050, December 2003.

[2] L. Chen, J. Xu, I. Djurdjevic, and S. Lin, “Near Shannon-Limit Quasi-Cyclic Low-Density Parity-Check Codes,” IEEE Transactions on Communications, Vol. 52, pp. 1038-1042, July 2004.

[3] M. O’ Sullivan,J. Brevik and R. Wolski, “The Performance of LDPC Codes with Large Girth,” Proc. of the 43rd Annual Allerton Conference; Communication, Control and Computing, September 2005.

[4] M. P. Fossorier, “Quasi-Cyclic Low-Density Parity-Check Codes From Circulant Permutation Matrices,” IEEE Transactions on Information Theory, Vol. 50, pp.1788-1793, August 2004.

[5] M. O’ Sullivan, “Algebraic Construction of Sparse Matrices with Large Girth,” IEEE Transactions on Information Theory, Vol.52, pp. 718-727, February 2006.

[6] Y. Kou, S. Lin and M.P.C Fossorier ,“ Low Density Parity Check Codes Based on Finite Geometries: A Rediscovery and New Results,” IEEE Transactions on Information Theory, Vol. 47, No. 7, pp. 2711-2736, November 2001.

[7] L. Jing, J. Lin and W. Zhu, “Design of Quasi-Cyclic Low-Density Parity-Check Codes with Large Girth” ETRI Journal, Vol. 29, No. 3,pp.381-389 June 2007.

[8] B. Ammar, B. Honary, Y. Kou, J. Xu and S.Lin,“Construction of Low-Density Parity-Check Codes Based on Balanced Incomplete Block Designs”, IEEE Transactions on Information Theory, Vol. 50, No. 8, pp.1257-1268, June 2004.

[9] J. Fan, Y. Xiao and K. Kim, “Design LDPC Codes without Cycles of Length 4 and 6”, Research Letters in Communications, Vol. 2008, Article ID 354137.

[10] S. Kim,J. No, H. Chung and D. Shin, “Construction of Protographs for QC-LDPC Codes With Girth Larger Than 12,” IEEE Transactions on Information Theory, Vol. 53, No. 8, pp.2885-2891.

[11] G. Malema, “Constructing Quasi-Cyclic LDPC Codes Using a Search Algorithm,” International Symposium on Signal Processing and Information Technology, Cairo Egypt, pp.969-973, December 2007.

[12] G. Malema, “Low-Density Parity-Check Codes: Construction and Implementation,” PhD Thesis, Electrical and Electronic Engineering, The University of Adelaide, Adelaide, Australia, 2007.

[13] Y. Wang, J. Yedidia and S. Draper, “Construction of High-Girth QC-LDPC Codes,”5th International Symposium on Turbo Codes and Related Topics, pp.180-185,Lausanne,Switzerland,September 2008.

[14] Z. Li , and BV.K. Vijaya Kumar, “A class of Good Quasi-Cyclic Low-Density Parity-Check Codes Based on Progressive Edge Growth Graph,” Proceedings of 38th Asilomar Conference Conference on signal, systems computing, pp. 1990 –1994, 2004.

[15] J. Campello, D.S. Dodha, and S. Rajagopalan, “Designing LDPC codes using Bit-Filling,” Proceedings of the International Conference on Communications, Vol.1, pp. 55-59, Helsinki, Finland, June 2001.

[16] X. Hu, E. Eleftheriou, and D. Arnold, “Progressive edge-growth Tanner Graphs,” Proc. IEEE GLOBECOM, Vol. 2, pp. 995-1001, San Antonio, TX, November 2004.

[17] S.Kim, J. No, H. Chung, and D. Shin, “Quasi-Cyclic Low-Density Parity-Check Codes with girths larger than 12,” IEEE Transactions on Information Theory, vol. 53, no. 8, pp. 2885- 2891,August 2007.

[18] G. Malema, “Construction of QC-LDPC Codes -qc_ldpc.m”,Mathworks http://www.mathworks.com/matlabcentral/fileexchange/34176.

[19] J.K. S Lee, B. Lee, J. Hamkins, J. Thorpe, K. Andrews, and S. Dolinar, “A Scalable Architecture for Structured LDPC Decoder,” IEEE International Symposium on Information Theory, pp.292-295, Chicago, IL, July 2004.

[20] K. Andrews, S. Dolinar, D. Divsalar, and J. Thorpe, “Design of Low-Density Parity-Check Codes for Deep Space Applications,” IPN Progress Report, pp.42-159, November 2004.

[21] M. Luby, M. Mitzenmachen, M. Shokrollahi, and D. Spielman, “Efficient Improved Low-Density Parity-Check Codes using Irregular Graphs,” IEEE Transactions on Information Theory, Vol. 47, pp.585- 598, 2001.

[22] D. MacKay, S. Wilson, and M. Davey, “Comparison of Constructions of Gallager Codes,” IEEE Transactions on Communications, Vol. 47, pp.1449-1454, October 1999.

[23] S. Johnson and S. Weller, “A Family of Irregular LDPC Codes with Low Encoding Complexity,”IEEE Communications Letters, Vol. 7, No. 2, pp.79-81, February 2003.

[24] G. Ritcher and A. Hof, “On a construction method of irregular LDPC codes without small stopping sets,” IEEE International Conference on Communications, Vol. 3, pp. 1119-1124, Istanbul, Turkey, June 2006.